



Logic & Fraud: Cross-Platform and Immortal

Garrett Held and Kevin Stadmeyer

Agenda

What logic and fraud vulnerabilities are

Why this vulnerability is different and why counter-measures aren't effective

Logic and fraud in the cloud and mobile

Real-life logic flaws

From exploit to design

Conclusions

What are logic and fraud vulnerabilities?

Logic Vulnerability

- Based on business rules or rules about how the application should behave
- May use no malicious characters, but cause the application to behave in an unexpected manner
- Every rule that should be enforced is also an attack vector
 - Example: E-commerce shopping cart

What are logic and fraud vulnerabilities?

Fraud Vulnerability

- Application logic is twisted for financial or personal gain
- Attack vector used to deceive
- May be a subset of a generic vulnerability, such as authentication and authorization problems, but does not have a universal attack pattern

What are logic and fraud vulnerabilities?

What do they look like?

Why these vulnerabilities are different

Classic Vulnerabilities

- Buffer Overflows
- Distributed Denial of Service
- Man-In-The-Middle

Why these vulnerabilities are different

Classic Vulnerabilities - Solved

- Buffer Overflows
 - String libraries: strncpy or strcpy
- Distributed Denial of Service
 - Use scalable, abstracted infrastructure: The Cloud
- Man-In-The-Middle
 - Encryption: Combine CA's and Cert-pinning and Convergence

Why these vulnerabilities are different

Classic Web Vulnerabilities

- SQL Injection
- Cross-Site Scripting
- Cross-Site Request Forgery

Why these vulnerabilities are different

Classic Web Vulnerabilities - Solved

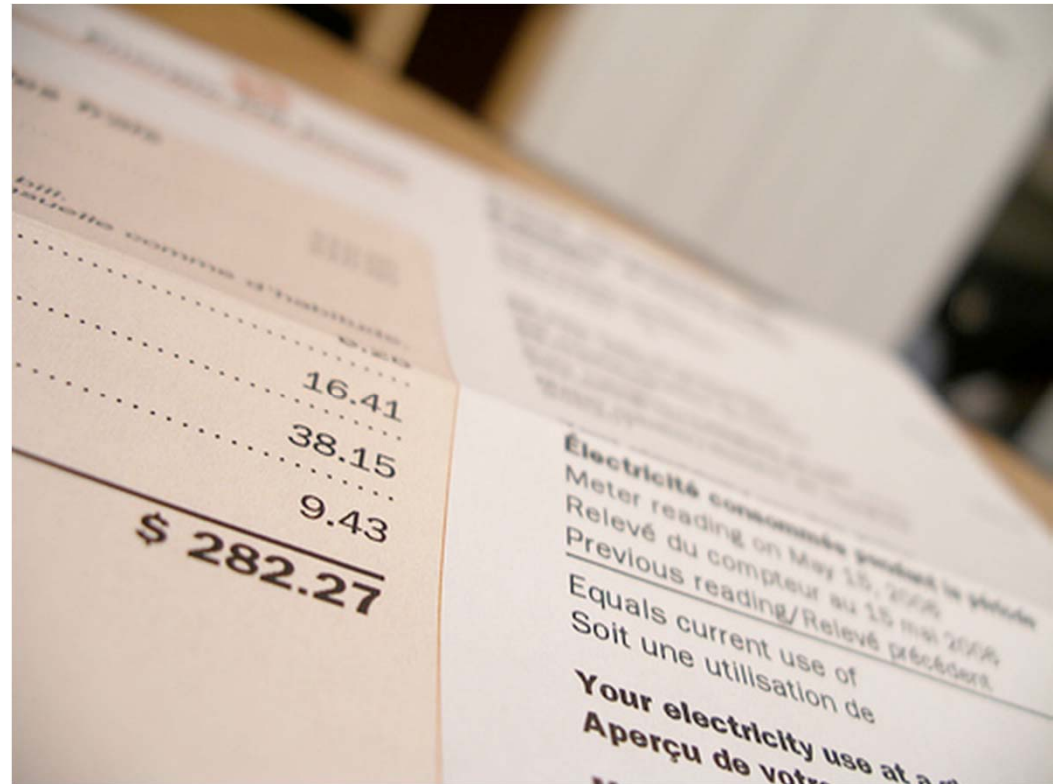
- SQL Injection
 - Parameterized Queries or Stored Procedures
- Cross-Site Scripting
 - Encode user-supplied output using built in Libraries
- Cross-Site Request Forgery
 - Include a framework-provided token in a hidden field

Fraud, IDOR, and the Cloud

Cloud-Based Bill Pay Application

1. Users cannot modify or delete other users payment information.
2. Strong input sanitization controls.
3. Stored numbers are masked.
4. Numbers never returned.

How do we commit fraud?



<http://www.flickr.com/photos/brendanwood/>

Fraud, IDOR, and the Cloud

Identifier was somewhat predictable.

When paying, request could be intercepted and identifiers swapped.

Looks correct to IDS and input checks...



<http://www.flickr.com/photos/aforero/>

Fraud, IDOR, and the Cloud

What went wrong?

Cloud or not, it doesn't matter

Authentication is required at EVERY step

You Ordered WHAT?

Online Store

Users without accounts can check order status

Required: Zip code, Order Number

Secure?

You Ordered WHAT?



JSON Request: What zip code for this order?



JSON Response: Oh, it's 12345



Do they match? Ok, now I'll send both



Here's your order information



You Ordered WHAT?

Bet I could script that!

Order numbers were predictable, may increase by two or three since every cart creation was an order number

```
For x=1; x < last_order_number; x++  
  last_name = Request using order number  
  order_info = Request using order number, last name
```



Looks good to
the IDS

Pulled several hundred orders as a proof of concept, hope you didn't order anything embarrassing!!

You Ordered WHAT?

Our thoughts

This will occur again in mobile applications wanting a rich user experience and not thinking communications are observed.

IDOR (Indirect Object Reference) will become more and more common.

Free university degree?



<http://www.flickr.com/photos/andymiah/>

Easy Auth Bypass

Mobile Client Oopsy

Three parameters

Two reset password forms

One Big Problem

- Testing was performed on the web app.
- Secure
- /passreset.html

Easy Auth Bypass

Mobile Client Oopsy

- Testing was performed on the Mobile app
 - (in)Secure
 - /mobPassReset.html
-
- Guesses?

Easy Auth Bypass

Mobile Client Oopsy

- Web client passed the `__authen_id` parameter.
- This parameter was ignored (rightly so).
- Mobile reset page decided this was an oversight (wrongly so).

Easy Auth Bypass

Mobile Client Oopsy

- POST something like this:

```
POST /mobPassReset.html HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:2.0) Gecko/20100101 Firefox/4.0
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Proxy-Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: <changed to protect the guilty, but paying>
Pragma: no-cache
Cache-Control: no-cache
Content-Length: 127

pass=BadIdeaBears&passrepeat=BadIdeaBears&__authen_id=0001873463
```

Easy Auth Bypass



<http://www.flickr.com/photos/abbot45/>

Easy Auth Bypass



<http://www.flickr.com/photos/devcentre/>

Make Your Own Security Questions



<http://www.flickr.com/photos/justanuptowngirl/>

Make Your Own Security Questions

Question 1	A user can put any question here they want?
Answer	All answers the same
Question 2	A user can put any question here they want?
Answer	All answers the same
Question 3	A user can put any question here they want?
Answer	All answers the same
Question 4	A user can put any question here they want?
Answer	All answers the same
Question 5	A user can put any question here they want?
Answer	All answers the same
Question 6	A user can put any question here they want?
Answer	All answers the same

Make Your Own Security Questions



<http://www.flickr.com/photos/drachmann/>

Make Your Own Security Questions



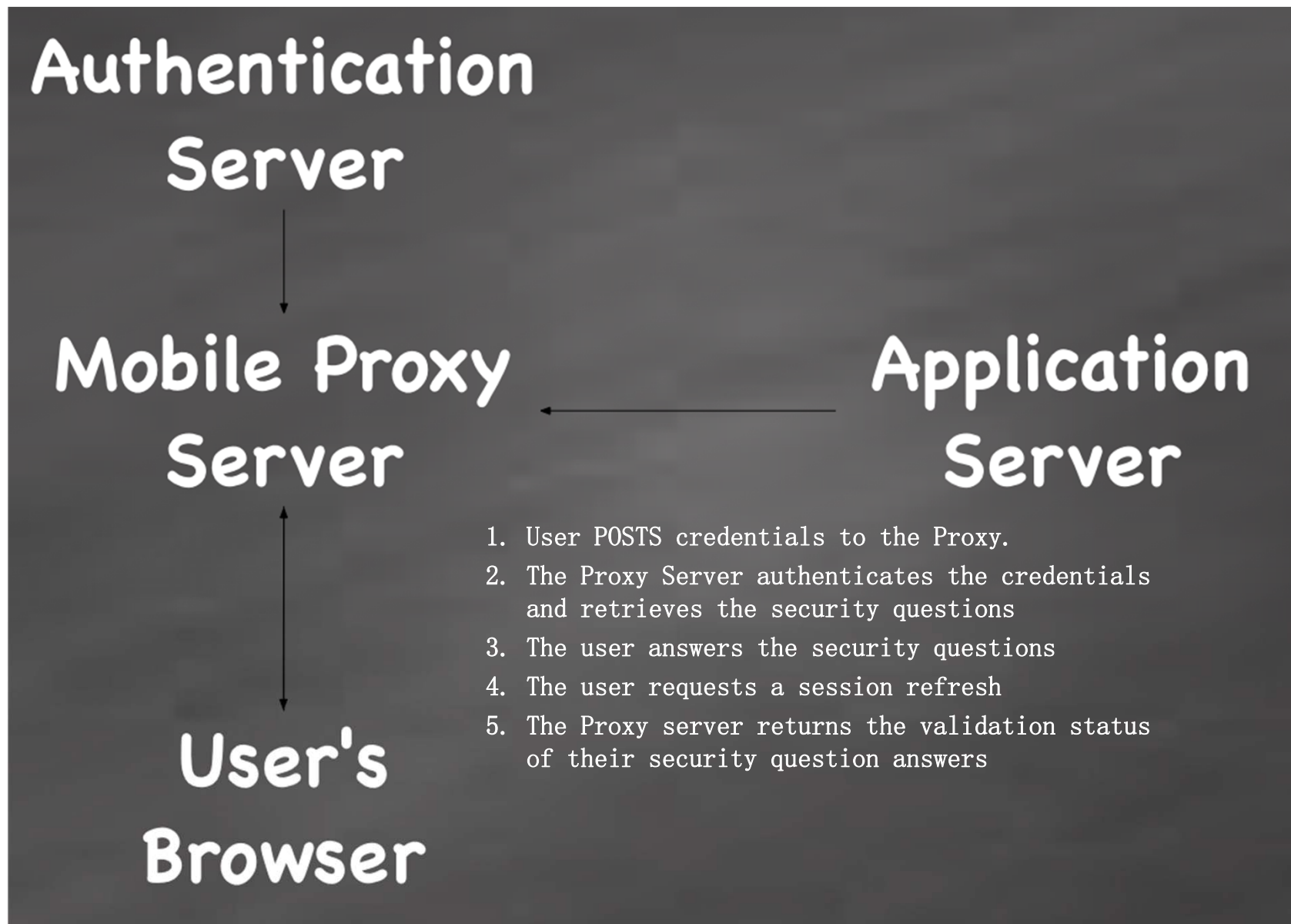
<http://www.flickr.com/photos/lumaxart/>

Harder (to find) auth bypass

Mobile Client Part Two

- Again a separate mobile site (see a trend?)
- This time only the login functionality was duplicated
- The application used on Security Questions

Harder (to find) auth bypass



Harder (to find) auth bypass

Mobile Client Part Two

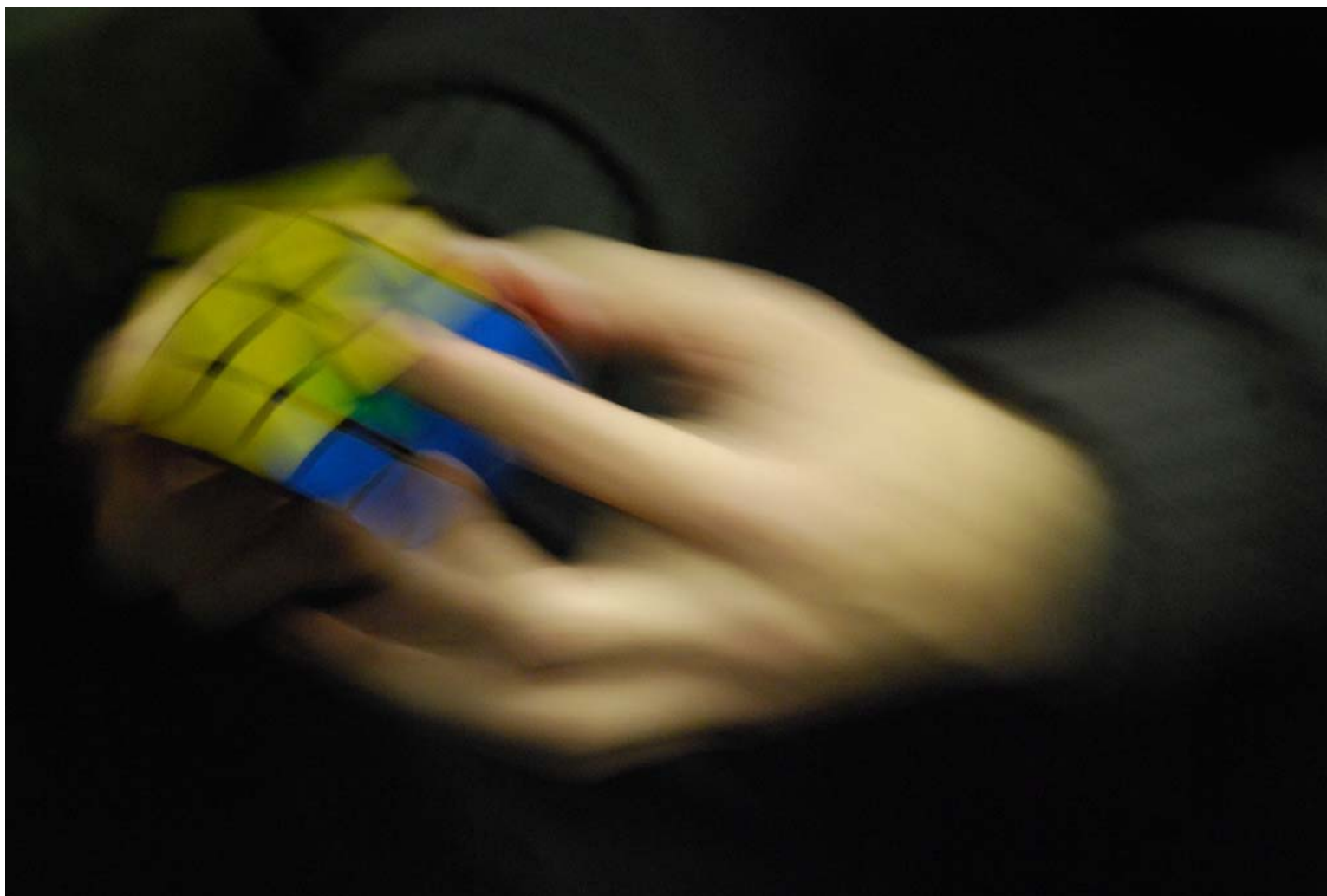


Harder (to find) auth bypass



<http://www.flickr.com/photos/horiavarlan/>

Harder (to find) auth bypass



<http://www.flickr.com/photos/ari/>

From exploit to design

Time Travel

What happened in these design meetings?

Assumptions!

From exploit to design

How do we guard against this?

Reverse the Definitions

- (1) Enforce business rules
- (2) Enforce order of steps
- (3) DO NOT trust the client
- (4) Special attention on IDOR



<http://www.flickr.com/photos/10047472@N08/>

From exploit to design

Let's design an application!

Restaurant Online Ordering

- (1) Our application takes the order
- (2) 3rd party card processor does the processing
- (3) Takes the information via a POST to their website
- (4) We pass an order number along that we can later reference and confirm the card was valid and charged



What assumptions should we consider?

<http://http://www.flickr.com/photos/wscullin/>

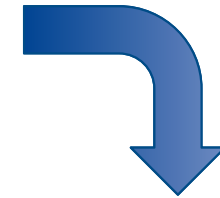
A(n) (almost) Free Lunch

Rushing to capture that Internet Money

- International Fast Food Chain
- No Standardized Online Order System
- Inexperienced Developers (used to developing in store POS software, not websites)
- Insecure Partners

A(n) (almost) Free Lunch

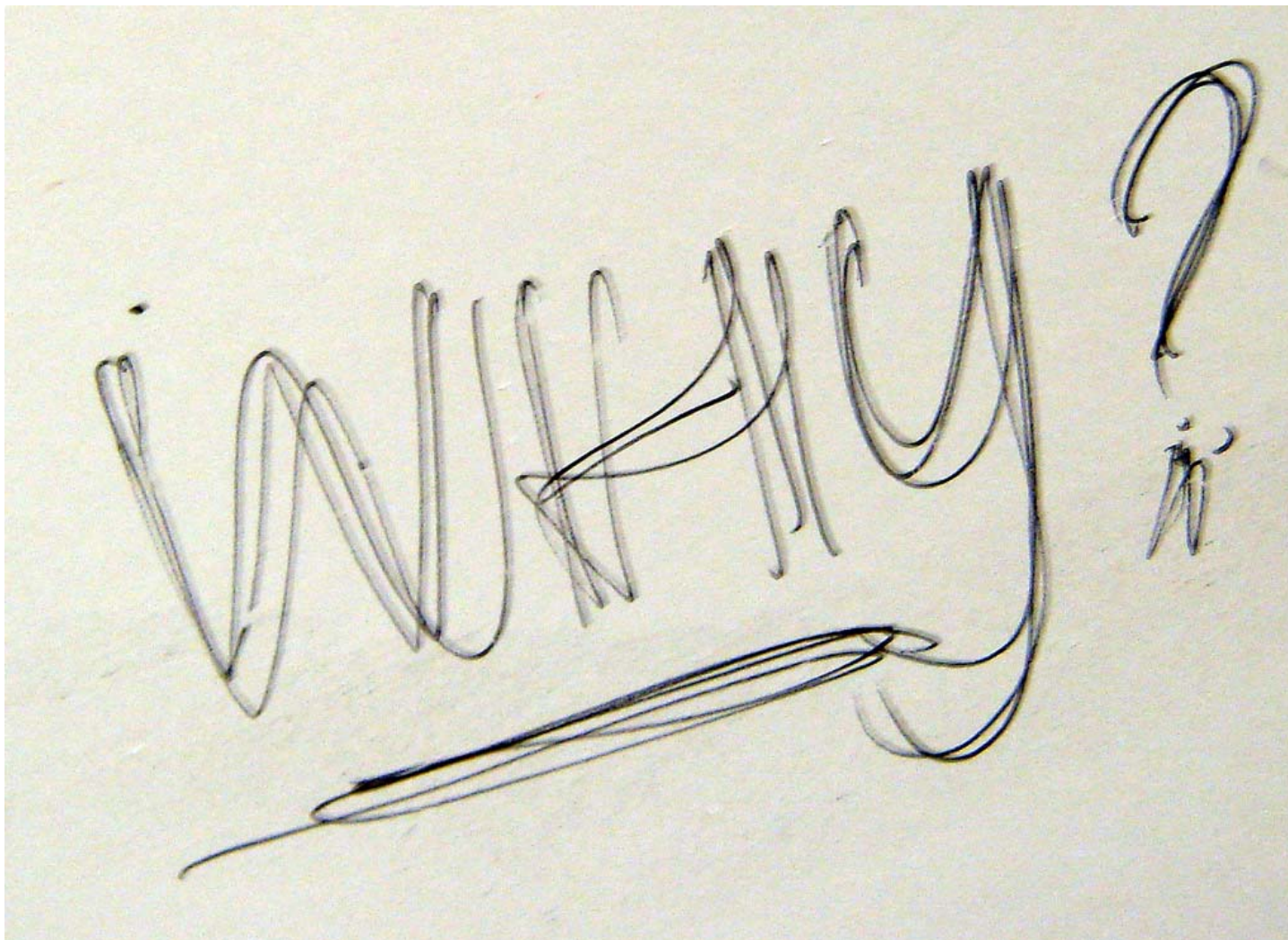
Minha Bandeja	
Valor do pedido:	3.50
Taxa de entrega:	6.00
Total do pedido:	9.50
QTD. PROD	VLR



My Tray	
Order value:	3.50
Delivery Rate:	6.00
Total Order:	9.50
QTD. PROD	VLR

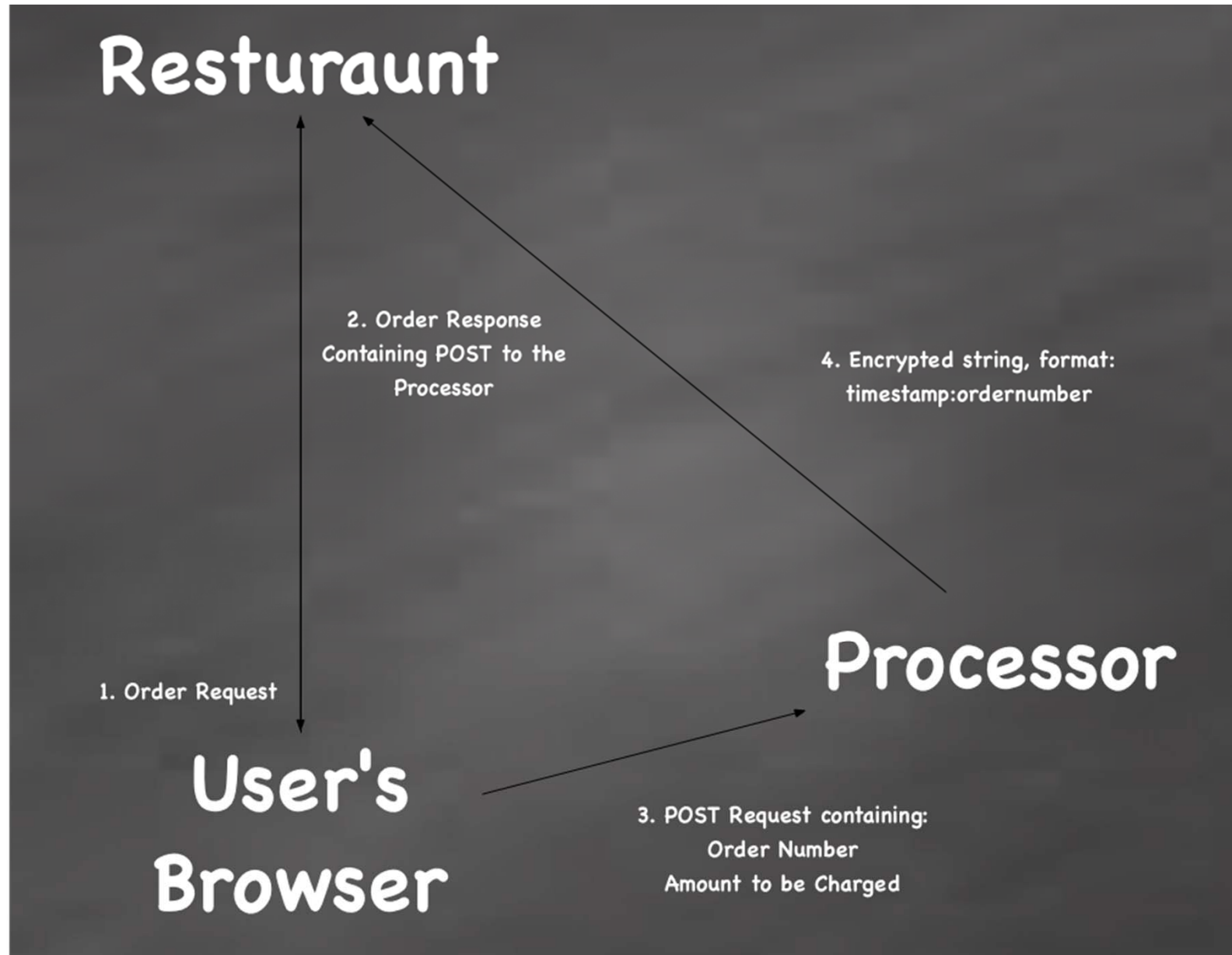
Actual Amount Charged: .50

A(n) (almost) Free Lunch



<http://www.flickr.com/photos/quinnanya/>

A(n) (almost) Free Lunch



A(n) (almost) Free Lunch



<http://www.flickr.com/photos/hradcanska/>

From exploit to design

Let's design an application add-on feature!

Patient Records

- (1) Patient record summary
- (2) Lookup **REQUIRES** both last name and patient number



What assumptions should we consider?

<http://http://www.flickr.com/photos/wscullin/>

When Queries Collide

Online Patient History

- Home
- Preferences
- Patient Query
- Billing History
- Billing Help
- Support
- Logout

Welcome to Online Patient History

From this site you can access any of the resources available via the links to the left.

Patient Payment History

Patient Number:
(16-digits)

Patient Last Name:

Submit ➤

When Queries Collide

Online Patient History

[Home](#)
[Preferences](#)
[Patient Query](#)
[Billing History](#)
[Billing Help](#)
[Support](#)
[Logout](#)

Welcome to Online Patient History

From this site you can access any of the resources available via the links to the left.

Patient Payment History

Patient Number:
(16-digits)

Patient Last Name:

Submit

Can we get this information elsewhere?

When Queries Collide

Online Patient History

- Home
- Preferences
- Patient Query
- Billing History
- Billing Help
- Support
- Logout

Patient Name Lookup

Phone Number:

OR

Patient Number:

Submit

From exploit to design

More we do to guard against this?

Understand Things!

- (1) Understand the framework!
- (2) Understand the enforcement!
- (3) Understand what your application trusts!
- (4) Understand your responsibilities!



<http://www.flickr.com/photos/10047472@N08/>

From exploit to design

Let's implement an application!

Password Reset (Commercial 3rd Party)



- (1) Password reset application for Windows
- (2) From Windows login you click a link that opens a restricted IE window (no address, no navigation)
- (3) Over SSL you answer security questions and then can reset your password.
- (4) No way to control where it navigates to, that is determined by a registry entry.

What assumptions should we consider?

<http://http://www.flickr.com/photos/wscullin/>

Attacking the Process

Don't pay attention to what the application does

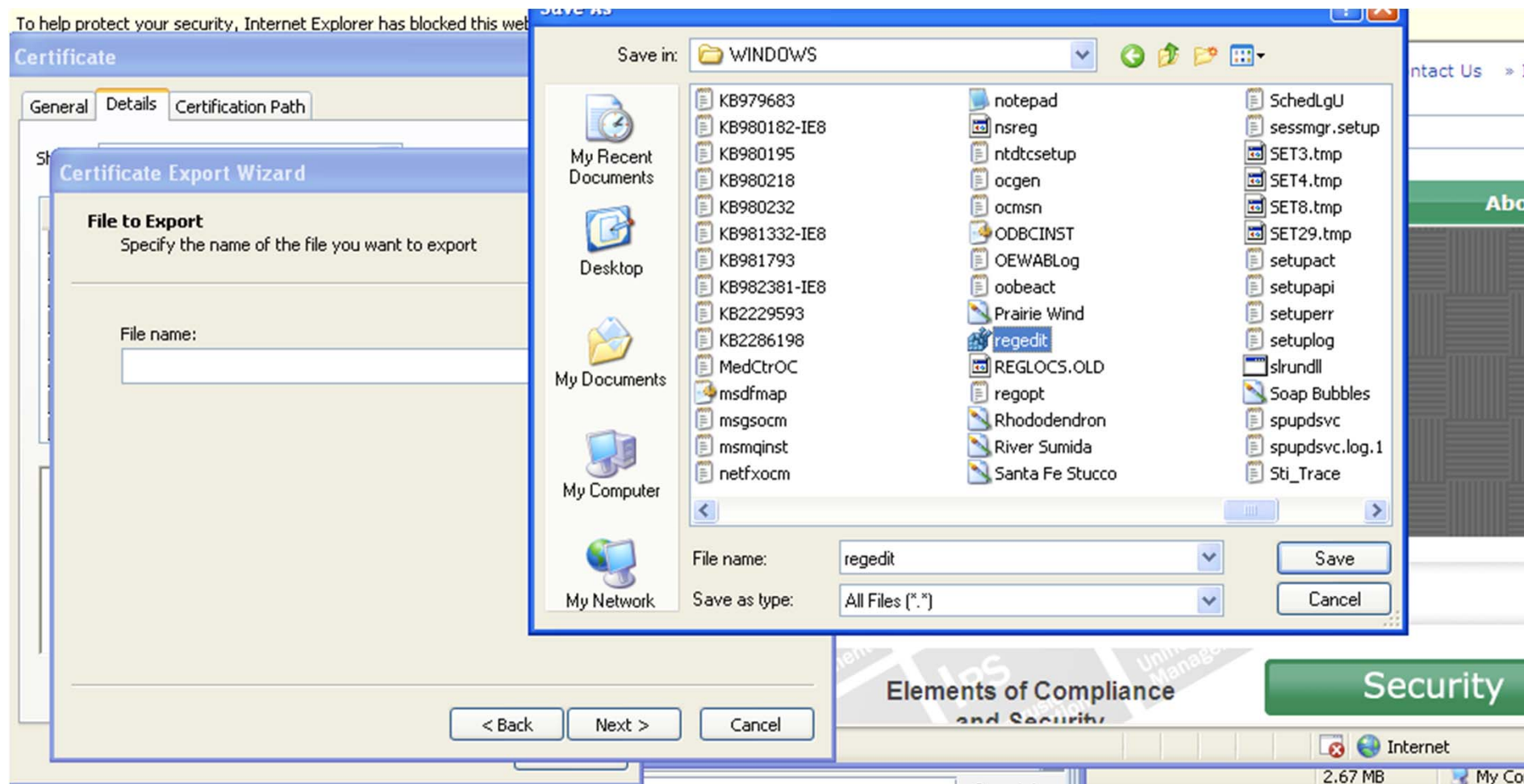
What might we do by disrupting the process?

Use good protections to do bad things?

Like logging Cross-Site Scripting that is executed when the log is viewed (or a book is previewed in an online bookstore).

What about a nice warning from IE 6 about a bad certificate if we tried a man-in-the-middle?

Attacking the Process



<https://www.trustwave.com/spiderlabs/advisories/TWSL2010-007.txt>

From exploit to design

More we do to guard against this?

Understand More!

- (1) Understand the reactions outside your control!
- (2) Ask how the good security controls could be used for bad!

Conclusion

Understand your application

Don't make assumptions about cloud or mobile

Threat model the logic

Understand the environment

Map the information relationships